Visual design of coherent technology enhanced learning systems: a Few Lessons learned from CPM language

Thierry Nodenot, Pierre Laforcade, Xavier Le Pallec

Visual Design of coherent Technology-Enhanced Learning Systems: a few lessons learned from CPM language

Thierry Nodenot, Laboratoire LIUPPA, Institut Universitaire de Technologie,
3 Avenue Jean Darrigrand, 64115 Bayonne Cedex, France
Tel: 33.5.59.57.43.53, Fax: 33.5.59.57.43.09
Email: Thierry.Nodenot@iutbayonne.univ-pau.fr

Pierre Laforcade, Laboratoire LIUM, Institut Universitaire de Technologie
52 Rue des Dr Calmette et Guérin, 53020 Laval Cedex 9, France
Tel: 33 2.43.59.49.13, Fax : 33 2.43.59.49.07
Email: pierre.laforcade@univ-lemans.fr

Xavier Le Pallec, Laboratoire Trigone, CUEEP,
Cité scientifique, Université de Lille 1
Tel: 33.3.20.43.32.89, Fax: 33.3.20.43.32.01
Email: xavier-le-pallec@univ-lille1.fr

Visual Design of coherent Technology-Enhanced Learning Systems: a few lessons learned from CPM language

Abstract. *Visual instructional design languages currently provide notations for representing the intermediate and final results of a knowledge engineering process. As some languages particularly focus on the formal representation of a learning design that can be transformed into machine interpretable code (i.e., IML-LD players), others have been developed to support the creativity of designers while exploring their problem-spaces and solutions.*

This chapter introduces CPM (Computer Problem-based Metamodel), a visual language for the instructional design of Problem-Based Learning (PBL) situations. On the one hand, CPM sketches of a PBL situation can improve communication within multidisciplinary ID teams; on the other hand, CPM blueprints can describe the functional components that a Technology-Enhanced Learning (TEL) system should offer to support such a PBL situation.

We first present the aims and the fundamentals of CPM language. Then, we analyze CPM usability using a set of CPM diagrams produced in a case study in a 'real-world' setting

INTRODUCTION

For several years, the IMS-LD specification (IMS, 2003b) has been the subject of converging theoretical and practical works from researchers and practitioners concerned with Learning Technologies.

The IMS-LD specification is now well documented (Hummel, Manderveld, Tattersall, & Koper, 2004; Koper *et al*., 2003; Koper & Olivier, 2004) and widely used for the semantic representation of learning designs. A *learning design* is defined as the description of the teaching-learning process that takes place in a unit of learning (Koper, 2006). The key principle in learning design is that it represents learning activities and support activities being performed by different persons (learners, teachers) in the context of a unit of learning. These activities can refer to different learning objects that are used/required by these activities at runtime (e.g., books, software programs, pictures); they can also refer to services (e.g., forums, chats, wikis) used to communicate and collaborate in the teaching-learning process.

Thus, IMS-LD is an Educational Modeling Language that provides a representation of the components of a learning environment in a standardized XML schema that can be executed by compliant e-learning platforms. According to the classification framework defined in (Botturi, Derntl, Boot, & Gigl, 2006), IMS-LD is an example of a 'finalist-communicative language': it is not intended to enable designers to produce intermediate models of the learning design being studied, nor to provide significant methodological support for designers to build a final representation complying with the IMS-LD specification.

Initially, designers had to use XML editors (like *XMLSpy*) to benefit from all IMS-LD expressive capabilities (levels A, B, C). *Reload*, a tree and form based authoring tool, was the first editor to significantly improve this situation. Chapter 2.10 of this handbook provides an extensive presentation of currently available IMS-LD compliant tools (Tattersall, 2007):
- LD-editors like *Reload* (Reload, 2005), *CopperAuthor* (CopperAuthor, 2005), etc.
- Visual tools to support practitioners in the creation of IMS-LD compliant designs by means of using collaborative pattern-based templates (Hernández-Leo et al., 2006).
- Authoring environments for IMS-LD designs like the *ASK Learning Designer Toolkit – ASK-LDT* (Sampson, Karampiperis, & Zervas, 2005).
- Runtime engines able to interpret a LD-scenario like *CopperCore* (Vogten & Martens, 2003).

- Learning Management Systems able to interpret LD scenarios: *dotLRN* (Santos, Boticario, & Barrera, 2005), *LAMS* (Dalziel, 2006), *Moodle* (Berggren et al., 2005), etc.

However, standards like IMS-LD (2003) and IEEE LOM (2002) start from the principle that even though learning theories are not pedagogically neutral, neutral reference models and standards can still be designed: 'The aim is not to set up a prescriptive model but an integrative pedagogical meta-model which is neutral since it models what is common with any pedagogical model' (Koper, 2001); this assumption promotes the concept of de-contextualized learning objects that can be specified once, and then reused to design Learning Scenarios relying on instructivist (acquisition metaphor) or constructivist (knowledge creation metaphor) principles.

This chapter proposes another way to address the design of learning scenarios. On the one hand, we consider that socio-constructivist learning scenarios must be designed in context. On the other hand, we think that even the final results of an Instructional Design (ID) process should clearly state the mapping between the contextualised activities specified by designers and the functionalities provided by a given Learning Management System (LMS).

In the first section, we present various on-going research work focusing on languages defined to help designers represent and share ideas about a learning scenario under study. Such languages are called 'generative-reflective languages' in (Botturi, Derntl et al., 2006). The second section introduces CPM (Cooperative Problem-based Metamodel) language, a visual design-language focusing on the design of Problem-Based Learning (PBL) situations; we present its syntax and semantics that rely on UML language. Then, we try to understand CPM usability from an analysis of a set of CPM diagrams produced in the framework of a real-world case study. This study illustrates CPM language expressivity; it also states that even though designing PBL situations with CPM notation remains a complex knowledge engineering activity, good practices can concretely improve designers' efficiency and confidence. Finally, the concluding section summarizes both CPM characteristics and proposals for improvement.

BACKGROUND

In this section, we only focus on current research work that could lead practitioners (teachers, educators, designers) to considering ID languages as adequate tools to explore their problem-spaces, not only to share ideas within a design team, but also to prepare the implementation of coherent Technological Enhanced Learning Systems.

Situated learning presupposes that meaning is both incorporated within the learning design as well as being prone to interpretation and shared understanding (Stahl, 2006): '*a blind spot of activity-centered models is their missing ability to describe the relation between the program (the learning design) and its context*'(Allert, 2004).

Thus, modelling coherent social systems for learning requires going beyond selecting and sequencing activities and resources; but also deciding and documenting for what purposes they are being used. This means that roles and activities are to be represented and assessed in context (Derntl & Hummel, 2005). With this purpose in mind, (Allert, 2005) introduces the concept of Second-Order Learning Objects (SOLOs) which are resources that provide and reflect a strategy (generative strategy, learning strategy, problem solving strategy, or decision-making strategy). SOLOs provide means for structuring information or modelling certain aspects of the real world: they represent sets of interrelated concepts that can be used to

describe the domain of concern. The use of different SOLOs will thus allow a designer to look at a system from different points of view (e.g., organisationally, structurally, and from social points of view).

(Pawlowski, 2002; Pawlowski & Bick, 2006) introduce the Didactical Object Model (*DIN)* which extends the aims of current Educational Modelling Languages by introducing specifications for contexts, experiences and acceptance. The concept of reusability is, in this case, extended since it should be possible not only to share scenarios as technical specifications but also to exchange didactical expertise about such scenarios (from the knowledge of their context of use, of concrete experiences reported by the actors involved in its use).

(Schneemayer, 2002), (Brusilovsky, 2004) and (Paramythis & Loidl-Reisinger, 2004) extend the context notion to the environment context which clarifies the real characteristics of the LMS (or any other software) from which the learning situation is being exploited. This leads to an approach for the engineering of learning situations aiming to specify the learning situation together with the LMS which will later enable students to learn from this situation.

Works of (Botturi, 2003; Botturi, Cantoni, Lepori, & Tardini, 2006) promote the adaptation of fast prototyping for the specific issues of e-learning project development with very particular stress on human-factor management (*i.e.*, the *eLab* model). They developed a visual design language called E^2ML (c.f. chapter 2.2 of this handbook) to support fast prototyping to enable a developing interdisciplinary team to function (including educators and teachers). Outcomes of the language include better communication within the design team, availability of precise design documentation to evaluate designs and figure out agreed and more feasible solutions.

Despite having quite different objectives, the works that we have listed in this section (including those conducted in the framework of the IMS-LD initiative) share the fact that they address the complexity of ID. Developing future Technology-Enhanced Learning (TEL) systems requires an interdisciplinary team with both pedagogical and technical skills: communication and minimal agreement on means and ends are conditions for success within such a team.

From the point of view of teachers and educators, ID languages can be communication catalysts (Botturi, Derntl et al., 2006) if these actors feel that the concepts of the language are in tune with the characteristics of the learning situation to be described and will enable them to explore, document and share their design decisions with others.

On the one hand, (Allert, 2005) states that teachers and educators need dedicated languages which reduce complexity by reflecting instruction (and the process of ID) according to specified criteria (p. 41): *i.e.,* formalization, compatibility and interoperability criteria (IMS, 2003b) are to be considered since most educators are now aware that the introduction of technologies in Education has important consequences on any design process.

On the other hand, such instructional languages must not neglect didactics, which is the science of learning and teaching; even if in the domain of training (reproductive forms of learning), the learning design is often limited to the planning and sequencing of non-contextualized activities and resources. (Pawlowski & Bick, 2006) state that designing

situated-learning requires languages that can precisely describe the context and the dynamics of the tutoring/learning activities and resources.

Our work on visual ID languages started just before (Koper, 2001) published his first results on the Educational Modelling Language (the precursor of the IMS-LD specification). From the very beginning, we intended to propose a visual design language that could be useful for both educators and developers of TEL systems. From the point of view of educators, the language requirements were,

1. To enable designers to represent learning-tutoring activities in context.
2. To reduce complexity by reflecting instruction (and the process of ID).

In the following sections, we shall first present the characteristics of the language; then we shall study the language usability from an analysis of its use on 'real-world' case studies.

CPM LANGUAGE

CPM stands for **C**ooperative **P**roblem-based learning **M**etamodel. It is a visual design language that we developed at the LIUPPA Laboratory (Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour, France) as a specialisation of UML language. CPM language focuses on the design of Problem-Based Learning (PBL) situations. We decided to work on such a dedicated language because we consider with (Allert, 2004) and (Pawlowski & Bick, 2006) that,

1. Pedagogical metamodels are not neutral,
2. There is an important need for design languages that specifically address generative learning (learning in context, situated learning).

According to the ID Classification Scheme defined in (Botturi, Derntl *et al.*, 2006), it is a visual (notation level), layered (stratification level), semi-formal (formalization level) language promoting multiple perspectives (more than one view) upon the same entities. In the next paragraphs, we present the aims of the language and the information model captured by CPM language. Fundamentals of both its abstract syntax (the CPM metamodel) and its concrete syntax (the CPM profile) are then discussed. Finally, we briefly present three real-world case studies, which have enabled us to experiment on the usability of CPM language.

Aims of CPM language

Even though learning by doing activities promoted by a PBL scenario may seem to be natural activities, PBL situations must be scripted. In the context of PBL, the support focuses on mentoring, motivating, creating simulated crises, showing how failures result from poor communication and lack of foresight, identifying and promoting areas in which teams and individuals have to make progress. Thus, PBL is different from traditional instructional methods which emphasize the content: this means the main focus is on the learner and genuine problems (Norman & Spohrer, 1996). Guided by tutors who take only a facilitator role, learners are engaged in active and meaningful cooperative learning. They collaborate with each other by using tools to represent problems, to generate solutions, to discuss different perspectives, to lead experiments and simulations, or to write reports, etc. The driving force is the problem given, the success is the solution of it, and apprenticeship is a condition for success. Thus, the object of any PBL activity is an ill-structured problem under study and the expected outcomes of a PBL activity are (Miao, 2000),

- Acquiring knowledge and skills which can be transferred to solve similar problems at individual level.
- Constructing shared knowledge and promoting mutual understanding at group level.

To address such objectives, our challenge was to explore UML modelling capabilities for the PBL domain and to adapt the semantics of this language, when required, using meta-modelling techniques.

UML is a standard controlled by the Object Management Group (OMG) which is widely known as a design catalyst within teams of software developers (Costagliola, De Lucia, Orefice, & Polese, 2002), (Ferruci, Tortora, & Vitello, 2002). Readers needing a basic understanding of the UML language will find a useful introduction in chapter 2.4 of this handbook.

UML language can be used as a sketch, blueprint or programming language (Fowler, 2005). In sketch usage, developers use UML to communicate some particular aspects of the system being studied. In the blueprint usage, the idea is to build a detailed design for a programmer to use in coding software. Blueprints may be used for all the details of a system or the designer may draw a blueprint for a particular area. In programming language usage, developers draw UML diagrams that are compiled directly into executable code, and UML becomes the source code.

Our studies demonstrated that UML is too general to correctly address PBL domain and interdisciplinary issues (Sallaberry, Nodenot, Marquesuzaà, Bessagnet, & Laforcade, 2002). Yet, UML activity diagrams are explicitly considered in (IMS, 2003a) as useful formalisms to capture requirements and build learning specifications. A UML-based language proved to supply more support to the interdisciplinary team of developers by means of well known (but debatable) UML features: standard notation, communication power, gateway between models and implementation platforms including software components and services.

Thus, we developed CPM, a specialization of UML language for PBL which we implemented by means of a Profiling mechanism (OMG, 1999). This language addresses most of the design process, covering the different stages of conceptual and functional designing. This was a matter of differentiating two target audiences.

On the one hand, *educators and designers* use CPM language to draw models (similar to UML sketches) focusing initial requirements of a PBL situation including the PBL domain, situated roles of learners/teachers, learners skills, predicted obstacles which the educators want learners to overcome, goals and criteria for success within the PBL situation, resources available to learners, etc.

On the other hand, CPM language addresses *instructional engineers*. Their work involves designing a viable solution, in coordinating all the actors involved in the development team. Knowledge of UML is a prerequisite for such engineers who use CPM language to draw various models which capture different points of view or outlooks on the same PBL situation (pedagogical, structural, social, or operational). This set of models makes up the learning/tutoring scenario which can be planned (in terms of steps and learning/tutoring events) but cannot be totally predetermined at design time since PBL addresses generative learning (Allert, 2005). The blueprints they produce are expressed in terms of the concepts appearing in the sketches produced by educators, thus facilitating discussion and agreement.

CPM sketches and blueprints prepare the detailed design stage that involves mapping those agreed CPM models with Platform-Independent Models (PIM), e.g., IMS-LD

(Laforcade, 2004) or LMS abstractions (Renaux, Caron, & Le Pallec, 2005). Even though we implemented a toolset to generate Level A IMS-LD compliant models from our CPM models, abstractions of LMSs are our favourite Platform-Independent Models. The idea consists in mapping conceptual design models with components representing abstract views of the services provided by an LMS: such a mapping lead designers to use the CPM language in order to specialize and contextualize the services supplied by an LMS according to the specificities of the activities to be fulfilled.

The CPM information model

CPM relies on an information model depicted in Figure 1 (Nodenot, 2005). It is composed of three blocks:

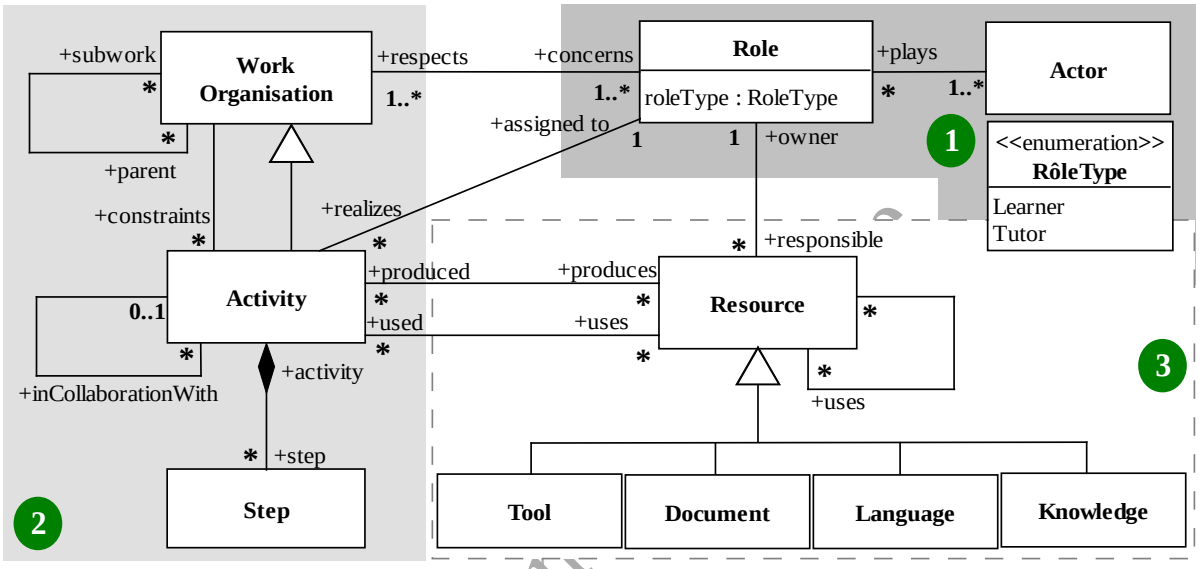


Figure 1: The CPM conceptual information model.

Block 1 (gray area at the top) deals with the modelling of the situated roles played by the very actors involved in a PBL situation. *Roles* can be assigned to individuals or to groups of actors. All roles do not imply the same knowledge and know-how; according to their learning goals and responsibilities, roles will often use specific *resources* to perform their learning/tutoring *activities*.

Block 2 (gray area at the left) deals with the *work organization* (rules that can constrain the way activities will be conducted by roles). This *work organization*, including collaborative work, can be decided by designers (learning scenario) or it can be in charge of the actors at runtime. When described at design stage, the organization rules may constrain the *activities* and *resources* at the learners/tutors' disposal. *Activities* can be further detailed in terms of *steps*, enabling designers to elicit the way important learning/tutoring events should be taken into account when they are raised at runtime.

Block 3 (white area at the bottom right) deals with the *resources* used by actors. *Knowledge* can represent activity prerequisites/post requisites, information about what can be learned from available documents, etc. A *language* is useful to the extent it forces actors to use a fixed set of vocabulary when they try to reach agreements in collaborative activities or when they are asked to describe what they know, what they would like to know, etc. *Documents* and *tools* represent contextualized artefacts enabling actors to conduct assigned activities.

The CPM toolset

From the CPM information model (to be compared with the IMS-LD Information model), we first built the abstract syntax of the language (the CPM metamodel) whereas its concrete syntax was represented through the CPM profile.

The CPM metamodel

To construct the CPM metamodel, an interdisciplinary team started with 35 concepts and divided them into two groups. First, concepts were selected which related to the necessity for the educators to produce a PBL situation's conceptual design (using terminology from works by (Develay, 1993) and (Meirieu, 1994) and includes notions like *Learning Goal*, *Obstacle*, *Success Criterion*, etc.). Then, several concepts were identified which are useful to describe a) the learning scenario (its structure and its dynamics) or b) the tool-environment provided to actors to conduct their learning/teaching activities. These concepts are borrowed as often as possible from the IMS-LD terminology (*e.g.*, *Activity*, *Activity-Structure*, *Role*, etc.). They are located in packages and sub-packages (see Figure 2): the *CPM_Foundation* (defined as a subset of UML 1.5) and the *CPM_Extensions* which adds the necessary concepts needed to describe PBL situations.

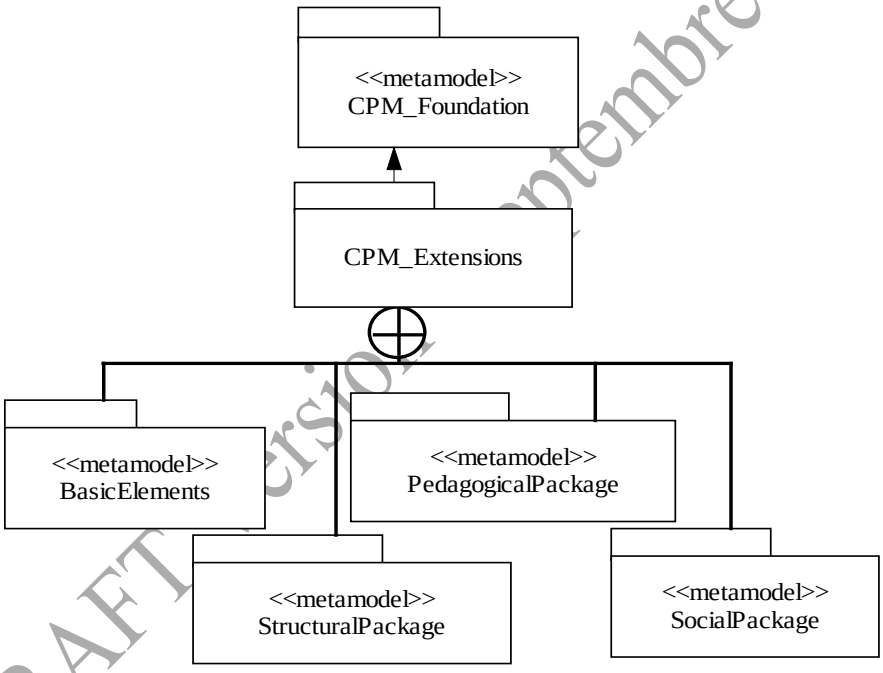


Figure 2: The packages of the CPM metamodel

Among CPM Extensions, cognitive concepts necessary to trace the learning/tutoring behaviours of the actors are included in the *PedagogicalPackage*. This package deals with information used to model the components of a PBLS: misconceptions of the learners, predicted obstacles that a teacher wants the learners to overcome, goals and success criteria of the PBLS, resources available to the learners, etc. The *StructuralPackage* includes concepts necessary to describe the PBL scenario and to break it down into simpler learning/tutoring activities. Lastly, the *SocialPackage* deals includes all the concepts necessary to manage co-operative work including sharing of resources and of learning/tutoring activities.

There are interconnections between the concepts within these packages. Figure 3 presents two extracts: on the left, a *Structural Package* extract and on the right a *Social Package* extract. Grey concepts refer to elements from the *CPM_Foundation* package (see UML 1.5).

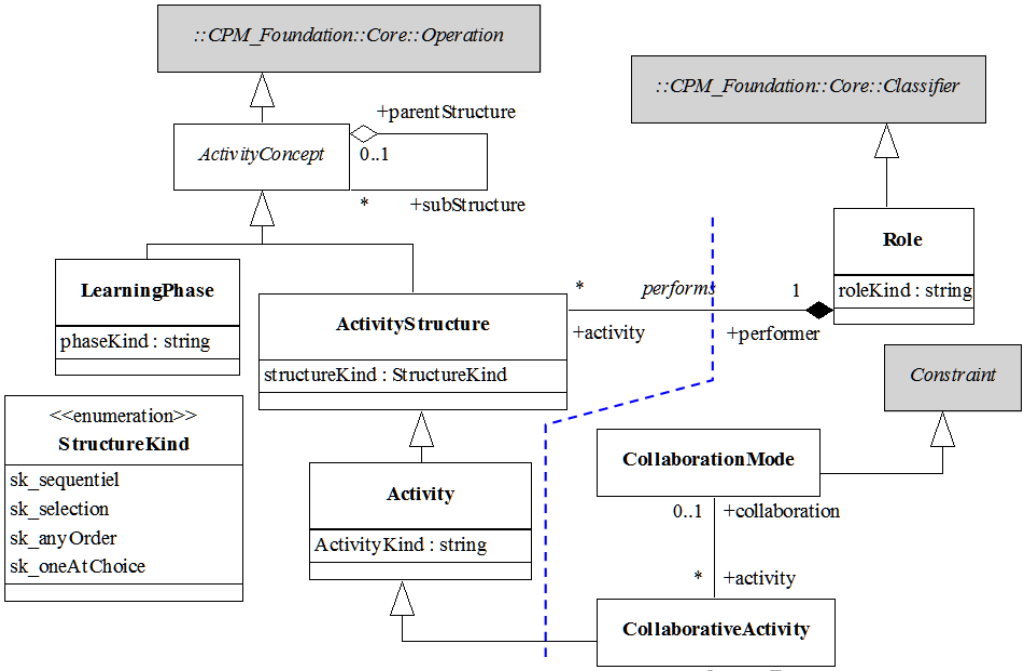


Figure 3: Interconnections between the concepts of the CPM metamodel packages.

- *ActivityConcept* particularizes the UML concept of Operation; it is a general concept to depict any hierarchy of activities.
- *Learning Phase* is used to sequence a Learning scenario; its semantics are close to the *Act* IMS-LD Concept, except that an IMS-LD *Act* can only be broken down into one and only one sublevel. Since it specialises the *ActivityConcept*, the *LearningPhase* concept can be used to describe a scenario with a hierarchy of acts including a hierarchy of scenes from which different roles will carry out particular activities.
- The *ActivityStructure* and *Activity* concepts are also specialisations of *ActivityConcept*; they respectively represent a group of activities and a particular activity assigned to one role. Activity Structures can be of different types (i.e.,. the *structureKind* meta-attribute).
- The *CollaborativeActivity* concept also specializes the *ActivityConcept*; the metamodel states that such an activity is performed by one and only one role (a role can be assigned to a group of concrete actors). Cooperation is not explicit in our metamodel since we decided to describe cooperation by means of role sharing and resource sharing (i.e., the CPM conceptual information model presented in Figure 1).

The CPM profile

To enable designers to draw diagrams that are consistent with such a metamodel, we implemented the CPM profile. A profile uses the extension mechanisms of UML in a standardized way, for a particular purpose. It merely refines the standard semantics of UML by adding further constraints and interpretations that capture domain specific semantics and modelling patterns.

Like any UML profile, the CPM profile promotes *Stereotypes* which are defined for each specific meta-class of the UML metamodel. Thus, for each concept of the *CPM_Extensions* package, we defined a particular stereotype attached to a specific UML meta-class (the Base meta-class) which the CPM concept directly or indirectly particularises. We also defined alternatives which are other UML meta-classes to enable designers to use a CPM concept in alternative UML diagrams than those suited to its Base meta-class. For example,

- a *Role* is a stereotype defined for the *UseCases::Actor* meta-class (i.e., figure 5) (a UML actor is something or someone who supplies a stimulus to the system operations).
- But we also promoted alternative meta-classes (i.e., figure 6): *ActivityGraphs::Partition* (to enable designers to use the CPM *Role* concept in UML activity-diagrams), Core::Classifier (to enable designers to use the CPM *LeaningPhase* concept in UML Class Diagrams).

This mechanism which was already used in (OMG, 2002a) means that *ActivityGraphs::Partition* and Core::Classifier are proxy notations of the *UseCases::Actor* meta-class.

Icons are associated to stereotypes to reduce the designers' cognitive load and to enhance visual appropriation of the CPM models. **Tagged values** are attached to the different stereotypes; they represent meta-attributes (e.g., *phaseKind*, *structureKind*, *roleKind*, etc.) of the *CPM_Extensions* concepts.

Stereotype	Metaclass	Icon
Activity	**Core::Operation** ActivityGraphs::ActionState ActivityGraphs::SubactivityState UseCases::UseCase Core::Classifier	
LearningPhase	**Core::Operation** ActivityGraphs::ActionState ActivityGraphs::SubactivityState UseCases::UseCase Core::Classifier	
Role	**UseCases::Actor** ActivityGraphs::Partition Core::Classifier	

Figure 4: An extract of the stereotypes provided to designers by the CPM profile.

We provided designers with an authoring environment supporting CPM language. This was developed alongside the *Objecteering/UML* CASE tool. This prototype allowed us to verify the coherence between the CPM profile entities (concrete syntax) and the CPM metamodel meta-types (abstract syntax). It also enabled us to store complete case studies (*e.g.*, the SMASH case study) as well as reusable design patterns in the Objecteering shared repository. The current release of this CPM language is available within a module that can be integrated in and used with the free-of-charge-version of the *Objecteering/UML Modeler*.

In the next sections, we shall denote a CPM stereotype with the << >> symbol (e.g., the <<activity-structure>> stereotype. A UML metaclass will be highlighted in italics (e.g., the *ObjectFlowState* metaclass). For the purpose of the case studies that we shall be presenting here, model elements which are instances of the CPM stereotypes will appear in italics (*e.g.*, the *Testimonies analysis* <<activity-structure>>).

Real world case studies designed with the CPM language

Chronologically, we started with the SMASH PBL situation that addresses 10 to 12 year old pupils who must piece together eye-witness accounts to identify the causes of a bicycle accident. We set up an interdisciplinary team including two teachers, two CPM specialists, two developers mastering the Moodle LMS. This team used CPM language to formalise the teaching/learning objectives, to imagine and to detail a cooperative learning scenario that could take advantage of available communication tools (chat, forum, etc.). The proposed scenario was then tested in real conditions during four half days within a classroom where groups of pupils assisted by their teacher had to cooperate according to the constraints of the

specified learning/tutoring activities (using dedicated resources—see figure 3). Dedicated tools (e.g., a dedicated e-whiteboard to help pupils share their understanding of the actors' spatial position when the accident occurred) were then developed to support learners activities; the scenario was then partly implemented for the Moodle LMS.

Proposed by (Vignollet, David, Ferraris, Martel, & Lejeune, 2006), the PLANET-GAME case study focused on the didactic transposition (see the initial requirements analysis in figure 2; see also the account in Chapter 2.7) of a learning game about astronomy. Assisted by a primary teacher, we used CPM language to describe the conceptualisation level that 12 year-old pupils can reach and, in the meantime, we selected different scientific properties of these planets: their distances from the sun, their day durations, their year durations, their compositions, their average temperatures, etc. This domain study led us to set more detailed learning/tutoring objectives from which we defined a learning scenario and tutoring strategies (Nodenot & Laforcade, 2006).

The GEODOC case study is an on-the-road project that lead us to formalize CPM scenarios putting the focus on learning/tutoring objectives dedicated to text comprehension as applied to Geography. Learning activities which we formalized with CPM language include actual and inferential questions about what is being read (identification and localisation of toponyms, topological identification, mapping-out of routes, *etc.*). This project investigates not only the specialization of LMS services according to formalized learning/teaching scenarios, but also the use of on-the-shelf computational applications in relation with the taught domain (*e.g., Postgis* and *GoogleEarth*).

In the next section, we briefly present the script of a learning scenario and we refer to the figures denoting the CPM diagrams produced in the course of the design of such a scenario. This will help us give concrete expression of the lessons learned from CPM language.

The *Act2* of the SMASH PBLS: What is this scenario about?

During *Act2* (i.e., the IMS-LD terminology), learners (who were previously divided into different groups) have to analyse allocated testimonies. While some groups (that is, *Investigator role 1 to 3*) have access to a limited set, others can read the full set of testimonies (i.e., *Investigator role 4*). The scenario leads all groups (there are several concurrent groups playing *Investigator role 1 to 3* while a unique group of learners plays the *Investigator role 4*) to exchange information about what they learned/understood from the accounts of the testimonies (each group will produce a *belief graph*) and then to write a single *accident report* that all groups must finally acknowledge. The learning scenario is supervised by the *Session manager role* and by a tutor (i.e., *the PoliceChief role*) whose job is to help learners develop an exhaustive analysis of the available testimonies at their disposal.

From a pedagogical viewpoint, such scenario script encourages the groups of learners to confront their own ideas of road safety (knowledge, know-how, attitudes) with the safety rules promoted by Road Regulations (Highway Code).

In the subsequent text, the reader will find several figures produced with CPM language to specify the *Act2* learning scenario. The model elements produced during the design process were all stored in the repository provided by the *Objecteering* UML Case tool (i.e., Figure 5) from the set of CPM diagrams produced by the ID Team in charge of the project. Each model element stored in the repository can be used in several diagrams: use-case diagrams, class-diagrams, activity diagrams, state-machines diagrams, etc. Among the different diagrams that were produced in the course of this project, the following were chosen for this chapter:

- Figure 6 and Figure 7 describe the roles taken by the actors and the coarse-grain activities they performed during *Act2*.
- Figure 8 describes the resources that *Investigator role 1* can use and produce when performing their dedicated activities.
- Figure 9 details the sequencing of the different coarse-grain activities and the conditions that resources must fulfil to accept transitions from one activity to another.
- Figure 10 and Figure 11 detail the *Testimonies Analysis* <<activity-structure>>.

In the next section, we shall use these figures to elicit the lessons that we learned about CPM language usability. However, from the information given about *Act2* in this subsection, we strongly encourage the reader to begin by analyzing the semantics conveyed by this set of inter-related CPM diagrams.

LESSONS LEARNED FROM CPM LANGUAGE

This section presents the lessons we learned about the usability of CPM language to edit / produce a learning scenario. From the three case studies summarized above, we drew two important lessons:

- Although CPM adopts the jargon that many pedagogues and educational designers already use, producing a set of coherent CPM models for a given case study is still a complex activity.
- Even though most pedagogues are not able to produce a set of CPM coherent models by themselves, both pedagogues and developers can contribute to and benefit from such design models.

Several observations led us to formalize these lessons. To give concrete expression to these observations, we shall rely on CPM models from the SMASH PBL; we shall particularly focus on the Act2 learning scenario (the end of the previous section) leading learners to investigate the causes of a bicycle accident from a set of eye-witness testimonies.

Lesson U1: Although CPM adopts the jargon that many pedagogues and educational designers already use, producing a set of coherent CPM models for a given case study is still a complex activity.

During the conducted case studies, we noticed that designers encountered difficulties when seeking to organize efficiently the different kinds of model elements that they were eliciting at design time (see Lesson U1 Observation 1). From the analysis of encountered difficulties and observed solutions, we propose a structuring model, which proved useful to organize the different model elements under study within cohesive packages.

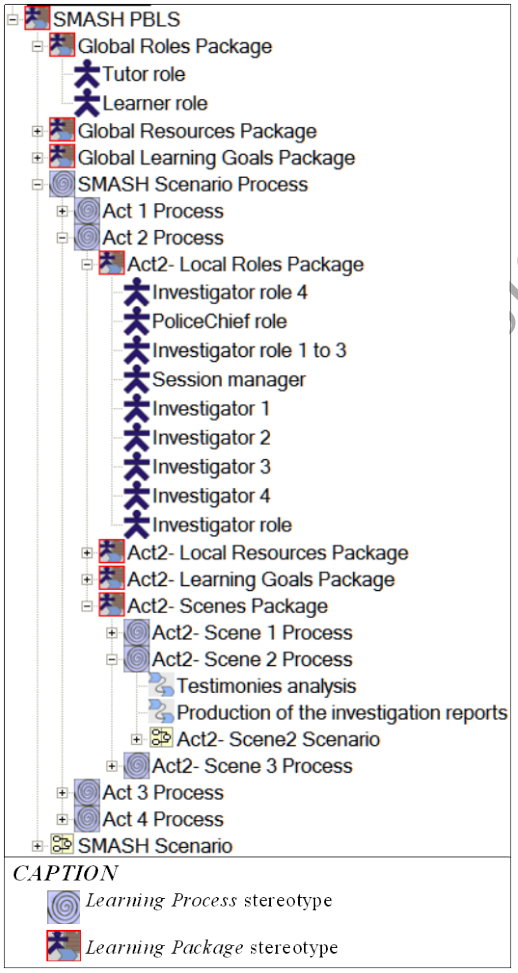
We also noticed (see Lesson U1 - Observation 2) that without human assistance, most educational designers did not know which notation was the most appropriate to represent their design intents. Yet, when the same educational designers gained experience about both the UML notation and about the CPM metamodel, most could produce expressive yet simple CPM diagrams.

Finally, Lesson U1 - Observation 3 shows that designers were sometimes frustrated because they were confusing CPM with a drawing tool: in particular, some did not clearly understand why the provided toolset (editors and wizards) considered as erroneous some diagrams whose model elements did not conform with the CPM metamodel.

Lesson U1 - Observation 1: relevant model elements must be conveniently organized by designers within packages. CPM diagrams must also be attached to packages.

Real world case studies that we specified with CPM language had in common that they could not be mastered by a single designer. All the modelling elements could not be represented in the same UML class diagram; learner, tutor roles, learning goals and success criteria had to be contextualized according to the steps of the learning process; both dynamics and structure of resources and activities had to be specified, etc. Relying on our experience in designing such case studies, we argue that in most cases, what is needed is an approach that structures the design of complex learning scenarios at different levels.

Packages are UML constructs which enable the grouping of model elements, making UML diagrams simpler and easier to understand. Packages themselves may be nested within others; they are depicted as file folders and may be Subsystems or Models. When we designed CPM language, we decided to provide designers with two stereotypes (see caption in Figure 5 which extends the *Package* metaclass: the *Learning Process* stereotype to break down the learning process into subprocesses and the *Learning Package* stereotype to group other model elements. In the course of our case studies, we learned efficient ways to exploit these stereotypes for organising model elements. For instance, Figure 5 describes the packages used in the SMASH PBLS:



This is a snapshot of the Browser which enables a designer to edit the SMASH Learning Scenario. At root level, experience led us to create three Learning Packages whose model elements are exploited by the *Learning Process* Package called the *SMASH Scenario Process.* At the bottom of the figure, worth noting is the *SMASH Scenario* denoted as an activity diagram used to generally describe how the different acts of the *SMASH Learning Process* are sequenced. The model elements (and graphical views) of these four acts are then detailed within the *SMASH Scenario Process*.

In the snapshot of Figure 5, the details of the *Act2 Process* were expanded. At this level, it will be observed that the package structure is the same as the one at root level: Act2 shows a *Local Roles Package*, a *Local resources Package*, a *Local Learning Roles Package*, and an *Act2 Scenes Package* which contains all the scenes within Act2. This structuring promotes the contextualisation of roles, learning goals, resources and learning activities. For example, the expanded *Act2 – Local Roles Package* shows different *Actor* stereotypes, which are model elements used during Act2 to specialize the tutor role and the Learner role (i.e., the *Global Roles Package*).

Figure 5: The SMASH PBLS browser.

It is worth noting that this approach is in tune with (Derntl & Motschnig-Pitrik, 2007) which encourages designers to elicit hierarchies of both learning goals and documents.

Lesson U1 - Observation 2: Among available CPM diagrams, designers must adequately choose those which can help them to produce some simple yet coherent perspectives of the relevant model elements.

First, let us recall that UML is a language enabling designers to describe an abstraction of a system that focuses on interesting aspects (models) and ignores irrelevant details. A perspective (view) focuses on a subset of a model to make it understandable.

Choosing UML to describe learning scenarios requires rethinking current uses and to elicit new uses of UML diagrams for dealing with the complexity of learning scenarios. From an educational point of view, a learning scenario is a system that must be described in terms of learning roles, learning goals, resources made available to the learners, learning and tutoring interactions / activities, events used to regulate learners' activities, etc.

From previous works (Sallaberry et al., 2002) we predicted the new uses of UML diagrams that CPM language encourages. As stated in the section devoted to the presentation of the CPM profile (see Figure 4), a CPM stereotype such as the <<Role>> Stereotype can extend either the *Actor* metaclass (to represent it in use-case diagrams, or the *Partition* metaclass (to represent it in Activity diagrams) or the Classifier metaclass (to represent it in Class diagrams).

During the course of our experiments, we noticed that designers (educators and computer-scientists) encountered two types of difficulties when trying to map their design intentions with available notation (those provided by the different types of diagrams available). First, most designers were inclined to start from a visual notation (e.g., the notation for class diagrams) and then tried using this specific notation to represent all perspectives of the model being studied, even if such a notation was not convenient for all aspects of the model. Second, we noticed that designers had questions about the notation they would be advised to use, particularly at the beginning of a learning scenario design process.

The case studies we have conducted provide useful answers to these difficulties. Let us focus on the intention, "role models involved in a learning scenario." If we consider the CPM information model given in Figure 1, designers should address different perspectives for roles. What are these? How are they involved in the *Work Organisation* that the learning scenario promotes? What are their responsibilities in the various (possibly collaborative) activities suggested to be performed in the scenario? What kind of resources do they exploit to carry out such activities? Applied to Act2 of the SMASH PBLS, Figure 6 and the following are CPM diagrams which focus on the different perspectives listed above.

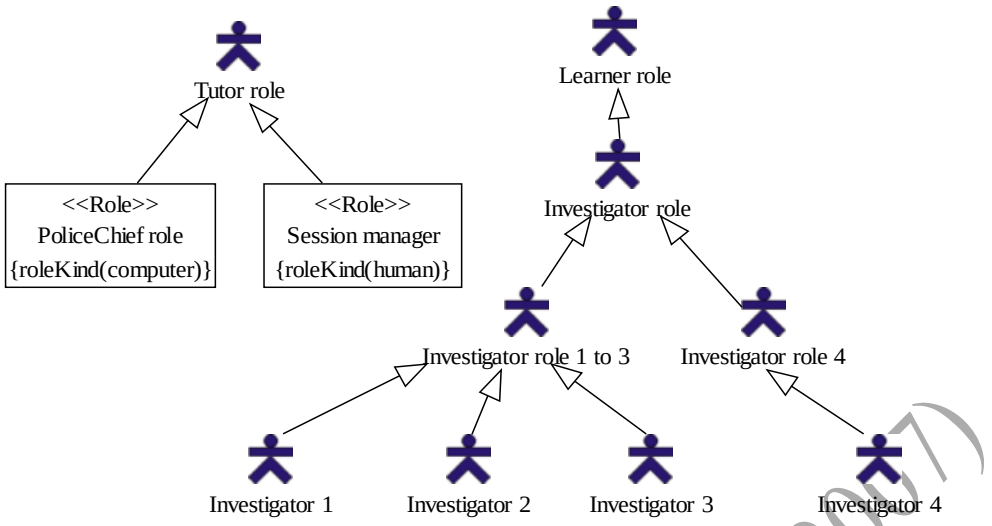


Figure 6: A class diagram representing a hierarchy of SMASH actors.

In Figure 6, SMASH roles specialise the *Class* metaclass. This class diagram shows that the *Learner* role and the *Tutor* role (from the *Global Roles* Package) were specialized to enable designers to denote all actors playing an important roles during Act 2. All roles are played by human beings except the *PoliceChief role (*we chose a detailed view of the *Tutor role* model element to make the *roleKind* tag-value visible). Figure 7 offers another perspective for these SMASH roles:

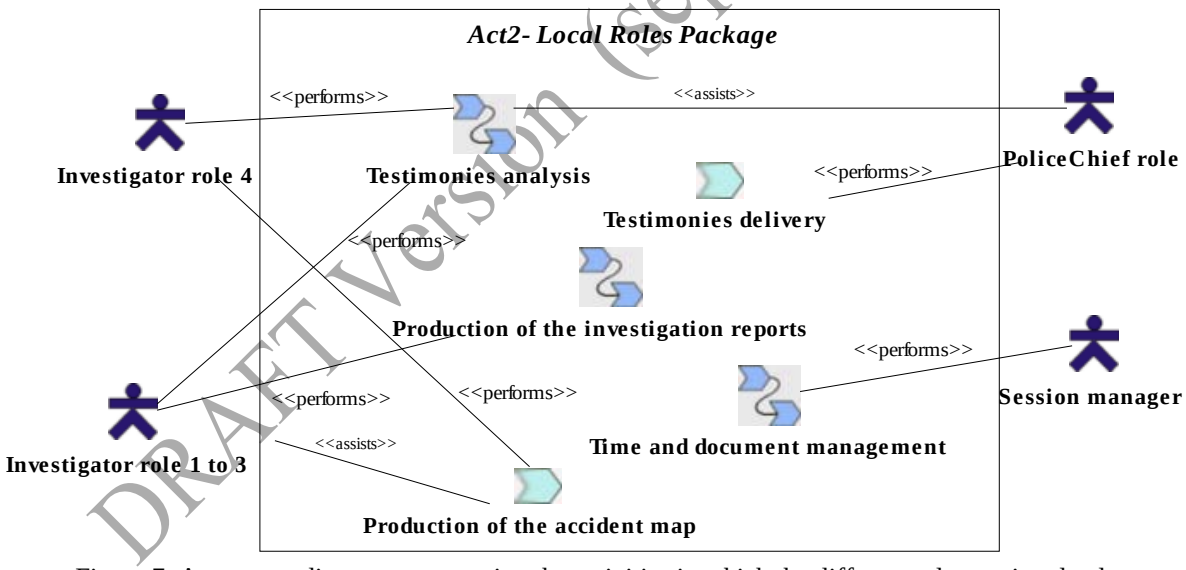


Figure 7: A use-case diagram representing the activities in which the different roles are involved.

In this use-case diagram, roles specialize the *Actor* metaclass. This perspective focuses on the activities carried out by roles during Act2. Each *role* either *performs a*ctivities or *assists* other roles performing those activities. Like in IMS-LD, activities that can be broken down into simpler ones (*e.g.*, *Testimonies analysis*, *Time and document managemen*t or *Production of the investigation reports*) are depicted with the stereotype <<Activity-structure>>.

Figure 8 is another class diagram which designers sketched to focus on the resources used and produced by each role during act 2 (there is a dedicated class diagram for each leaf role

that appears in Figure 6). Resources which are produced have the tag-value *output* while others have the tag-value *input*.

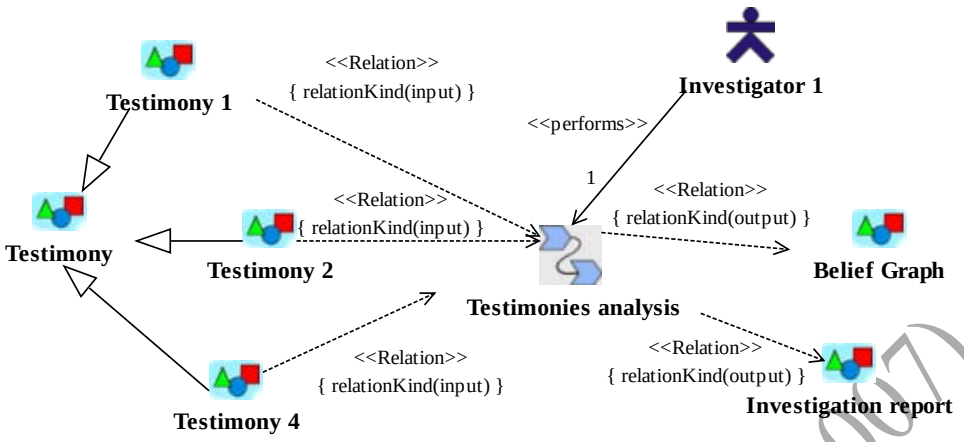


Figure 8: A class diagram describing the resources used and produced by the role Investigator 1.

The different figures provided in this section clearly show that the different perspectives provided to describe the roles in Act 2 are complementary (all of them can be reached from the model elements Browser presented in Figure 5). Other types of diagrams will be presented in Figure 10 (an Activity diagram) and in Figure 11 (a state-machine diagram) to respectively detail the *Testimonies analysis* model element and the Belief Graph model element that appeared in Figure 7 and Figure 8.

These figures also show that UML notations must be understood by designers to enable them to produce simple yet coherent perspectives of the learning scenario being studied. Table 1 provides a synthesis of the practices we noticed during our case studies. To build this table, we took into account only diagrams which appeared in the last version of the design produced for each of our case studies.

	Use
Activity Diagram	- External analysis of the learning Scenario - Description of collaborative activities - Internal analysis of activities and activity-structures
Use Case Diagram	- Activity cut-out - Role identification
Class Diagram	- Learning goal description - Role description - Resource description - External analysis of activities and activity-structures - Description of the concepts from the domain model
State Machine Diagram	- Description of the active classes (resources, roles, learning goals, activities)
Object Diagram	- instances from the domain model (concepts being studied, knowledge and know-how that learners must acquire)

Table 1: Best practices for CPM diagrams.

The reader may be surprised that we do not recommend the use of the Object Diagram for the definition of roles and of resources. In fact, experience led us to consider that concrete

roles appear only when the scenario is deployed on a platform (LMS) and used by concrete (groups of) learners. It is only at deployment time that the *Investigator role 1* stereotype is instantiated and played by concrete learners. And for similar reasons, the resources produced and used by *Investigator 1* are represented as classes (i.e., Figure 8) and not as objects.

Lesson U1 - Observation 3: To succeed in producing a perspective, designers must agree on both the UML notation and the CPM metamodel which both define the rules that the model elements in a CPM diagram must fulfil.

During our experiments, designers were at first surprised (and a bit confused) that they were constrained by both the rules of UML notations and of the CPM metamodel. On the one hand rules from the UML notations, they could not add, for example; any information about the timeline in the class diagrams being sketched. On the other hand, the CPM metamodel forced them to respect, for example, the following rule: when the <<activity>> and the <<resource>> stereotypes both extend the *Classifier* metaclass (i.e., the class diagram in Figure 8), connection links between such stereotypes must be of type <<Relation>> (the tag-value can either be *input* or *output*). Most designers did not understand such CPM rules, because they did not realize that the same stereotype (e.g., the <<activity>> stereotype) could represent different metaclasses when used in different types of diagrams. For example, in Figure 7, the *Testimonies analysis* model element extends the *UseCase* metaclass while, in Figure 8, it extends the *Classifier* metaclass (i.e., Figure 4 for the available metaclasses of the CPM stereotypes).

The three types of observations presented in this section show that designers need time to gain the necessary experience required to relevantly exploit the CPM language. Our experience also showed that educators can understand the meaning of a set of CPM diagrams but that the (semi) formal nature of CPM language could hinder some educators' commitment in producing such visual designs. They ask for cognitive assistance during the design process: since CPM editors do not allow free drawing, designers require some feedback enabling them to do some opportunistic productions: to-do lists, checklists, wizards, etc.

The first cognitive tools developed were contextual menus that could infer the metaclass to be used from the knowledge of both the diagram type and the stereotype chosen by the designer. In the framework of our latest project (the GEODOC case study), we also provided designers (educators and computer-scientists) with the best-practices of CPM diagrams and with a set of sample CPM diagrams for each design intent listed in Table 1. Our first experimental results show that such a design team was more efficient (time and design quality) than another team that did not have such documents at their disposal.

But it is already clear that our toolset is still a research prototype that proved expressive capabilities but cannot be distributed to an interdisciplinary team without care and human guidance. Even though the current state of research presented in this section can provide substantial support in understanding PBL scenarios, in designing and documenting new scenarios, it is clear that our approach is specified by rather technically oriented computer science people and a lot of work is still necessary to transform educators into CPM autonomous designers.

Lesson U2: Even though most pedagogues were not able to produce a set of CPM coherent models, both pedagogues and developers can contribute to and benefit from such design models.

Though educational expressivity of CPM diagrams, Lesson 1 pinpointed some difficulties encountered by designers who used the CPM toolset. In this section, we present some methodological principles which can help an ID team control the design process complexity.

In the course of the conducted case studies, we first observed that, at any level of the learning scenario analysis (conceptual design, functional design), designers might produce simple yet expressive CPM diagrams (i.e., Lesson U2 - Observation 1): it is a matter of focusing on one and only one perspective at a time.

We also noticed that a correct stratification of the learning scenario was important (i.e., *Lesson U2 – Observation 2*) to ensure a smooth transition between the perspectives drawn during learning scenario conceptual design and those drawn to address the functional design of a TEL system that could manage such a learning scenario at runtime.

Both observations will lead us to elicit a design process in tune with CPM language characteristics.

Lesson U2 - Observation 1: complexity of models can be mastered by designers using the following rule: Design only what is necessary for a given purpose and recognize overdesign.

Our experience is that most pedagogues can concretely draw various CPM diagrams if they keep in mind that each diagram should focus on one perspective that remains simple and expressive. Consider the *Testimonies analysis* model element which appears in Figure 7 and in Figure 8. None of these perspectives provides information about the activity sequencing planned during *Act2*. Adding such an information within Figure 7 is difficult since use-case diagrams are not suited to the description of activity sequencing: in general, UML specialists add OCL constraints (OMG, 2002b) to address such difficulty. Drawing another perspective focusing on such activity sequencing is much easier as stated in Figure 9:

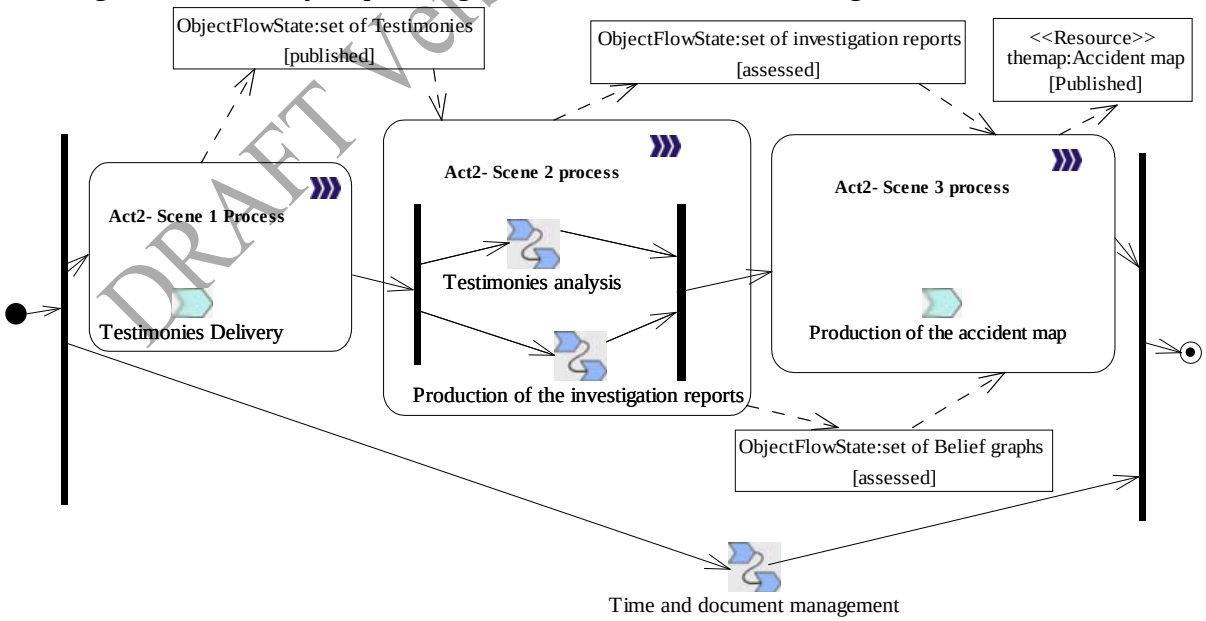


Figure 9: An activity diagram describing the sequencing of the activities performed during *Act2*.

In this figure, the reader will notice all activities and all activity-structures that already appeared in the *Act2* use-case diagram presented in Figure 7: these model elements are grouped together according to the scene during which they are performed by these actors. The

information flows between states as *ObjectFlowStates*: these represent some events that should be true either at the beginning (prerequisite) or at the end (post-requisite) of each scene. These different scenes (*e.g.*, the *Act2- Scene 2 process*) are structuring model elements that can also be easily located in our SMASH browser (i.e., Figure 5).

We consider that such a diagram can also illustrate what over-design means. At the conceptual design level where educators play the most important role, it would be useless to try to represent exception-handling in such a predicted learning scenario. At runtime, such a script can raise many exceptions (potentially meaningful for educators) that need to be managed (particularly those in relation with *the Time and document management* <<activity-structure>>). But adding exception handling in such a diagram would be likely to complicate the perspective and could mask the key ideas of the scenario, which were already spotted in Figure 9.

As a consequence, we consider that educators relying on CPM for conceptual design with CPM should strive for an 80% solution: at this stage, visual design should be used to represent the intermediate and then the final results of the design, thus providing means of communication between educators and computer scientists. All diagrams presented above are still intermediate results of design which helped educators clarifying and sharing their initial ideas.

CPM activity diagrams are other important perspectives to consider because they are a (natural) bridge between the use-case diagrams (which are useful to represent educational roles, goals and activities) and the class-diagrams (that developers need to implement required functionality on a learning platform). During our experiments, such diagrams represented an interesting communication trade-off between our business logic experts (educators and interaction designers) and Information Technology experts (software designers, learning platform specialists, etc.).

For example, Figure 10 is an activity diagram that details the *Testimonies analysis* <<Activity-structure>>.

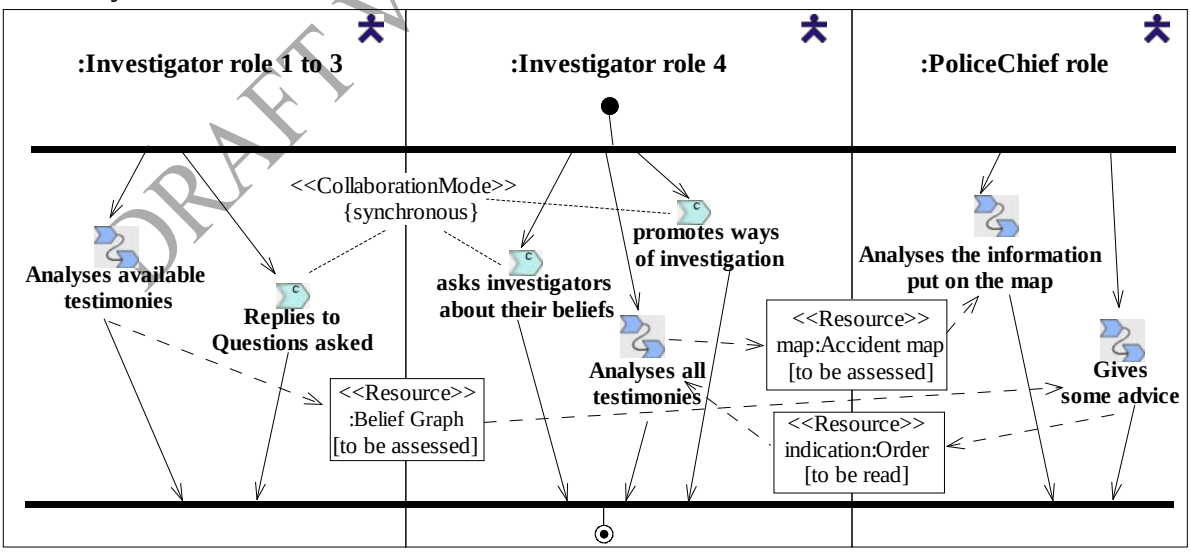


Figure 10: An activity diagram to represent the details of the *Testimonies analysis* activity-structure.

Three swimlanes are used to identify the specific activities performed by each role; these swimlanes are consistent with the roles assigned to the *Testimonies analysis* <<activity-structure>> in the use-case diagram presented in Figure 7. In Figure 10, we can notice that the

Testimonies analysis <<activity-structure>> exposes four activity-structures (*e.g.*, the *Analysis available testimonies* <<activity-structure>>) that can be further detailed using a top-down approach, some collaborative activities (e.g., *Replies to Questions asked* <<collaborative activity>>), some resources (e.g., *the Belief Graph* <<Resource>> to be assessed when it is updated by any real actor playing the <<role>> called *Investigator role 1 to 3*).

Figure 10 also denotes how designers can describe collaborative activities (i.e., activities with a *c* flag); in the scenario, *Investigator role 1 to 3* cannot initiate any synchronous conversation but this role can read information and answers questions asked by *Investigator role 4* (at implementation stage, and will lead developers to specialize a chat service according to these requirements).

An *ObjectFlowstate* denoting a <<Resource>> can be described with a UML State-machine diagram. For example, Figure 11 represents the lifecycle of the *Belief Graph* model element elicited in Figure 10:

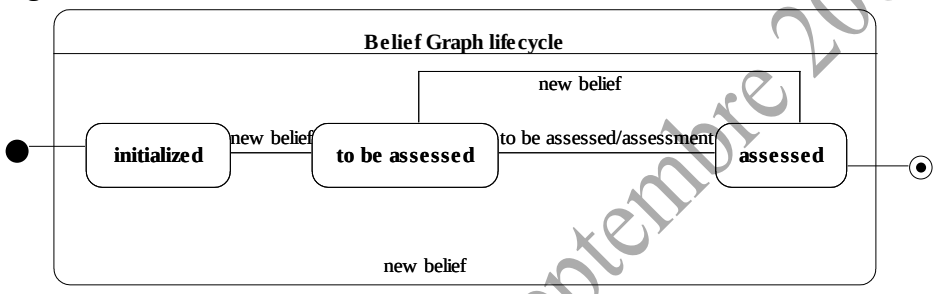


Figure 11: a state-machine diagram to represent the lifecycle of the *Belief Grap*h <<resource>>.

The underlying semantics is the following: each time an investigator adds a belief in his belief graph (*e.g.*, a representation of the following belief: "the white car bumped into the back of the bicycle"), the state of the belief graph changes to "to be assessed" (since the *PoliceChief role* is played by a machine—that is, the class diagram in Figure 6-, such a decision will entail particular design concern about the assessment process elicitation).

We noticed that educators encountered various difficulties when seeking to draw some CPM activity diagrams by themselves. It is true that these diagrams are not simple to create but they allow complex system/interaction processing to be represented efficiently. In order to get round this obstacle, we advised educators to produce a use-case diagram (in our example, a use case-diagram detailing the *Testimonies analysis* <<<activity-structure>>) for identifying the activities of interest and their relationships; Information Technology designers used such sketches for discussion purposes with them; and together they produced the final 80% solution presented in Figure 10. Interestingly enough, once this deadlock was broken, educators were able to go further in the conceptual design process.

From this set of observations, we learned that when using CPM diagrams for modelling a learning/tutoring scenario, it is important to capture the requirements at a high level of abstraction. Whatever the diagram, the perspective must remain simple. Such an approach allows designers to emphasize important model elements while hiding low-level processing details. Indeed, such details may even obscure the model's true purpose, which is,

a. to identifying key activities and dependencies,
b. to promoting exchanges and communication in the ID team.

This is particularly true when drawing activity diagrams. In our experiments, some of these proved to be potential deadlocks that frustrated most educators during the design process. Dedicated cognitive tools (wizards, to-do-lists, etc.) could probably give them more confidence; but we consider that the correct answer will rely on efficient communication in the ID team. With this in mind, sketches (even when they represent intermediate design results) can now play a central role in enhancing such communication.

Lesson U2 - Observation 2: CPM contributes to producing both stratified and multiple perspectives for a given learning scenario. This combination is a key-factor to enable a designer team to collaboratively determine the constraints under which a Technology-Enhanced Learning (TEL) system is to be designed.

UML is a widely accepted language to describe software systems. With the different perspectives of a TEL system that CPM offers, our profile adopts the same fundamentals (UML notation, UML semantics which we specialized with the CPM metamodel semantics) to also describe the educational context:
- At conceptual level, the language addresses the need to manage educational requirements effectively.
- At functional level, the language addresses the need to describe the required functionality of a TEL System in tune with such educational requirements.

Whatever the design level (conceptual *vs.* functional), it is very important therefore to communicate design decisions (and understanding) in an unambiguous form to all partners involved in the ID Team (including educators, Information Technology specialists and platform of learning developers).

In the previous subsections, we showed that CPM enables designers to produce multiple perspectives for a learning scenario. These perspectives favour coherent, unambiguous (within the limit of the UML semantics) but intelligible design decisions. The conducted case studies have also demonstrated that to reach such a goal, these multiple perspectives of a learning scenario should be correctly stratified. During the GEODOC case study, we noticed that, from the very beginning of the design process, some geographers were trying to map some educational goals with functionalities of the Geographical Information System viewer which they had been used to working with previously. Such design decisions were problematic because on the one side, educational goals had still to be further detailed and on the other, such a detailed analysis failed because the designers were mixing conceptual and functional model elements.

The main gains of a correct stratification are modularity and design simplicity (i.e., Lesson U2 Observation 1 in the previous subsection). Modularity allows easier adaptability when changing requirements; it also allows clear separation of the domains of trust. By starting with the most fundamental educational factors (conceptual design) and designing them to be contextually appropriate, we were able in the course of the conducted case studies to build successive layers design and eventually reach functional design.

Figure 12 is an activity-diagram which exemplifies the frontier between conceptual and functional design. In this figure, some activities denote a <<CPL>> stereotype that represents a functionality offered by concrete software components. Such components may be those provided by most Learning Platforms (*e.g.,* a quiz component, a lecture component, a forum

component, a whiteboard component, etc.) or they may be specialized components in relation with the domain to be taught (*e.g.*, a Geographical Information System viewer).

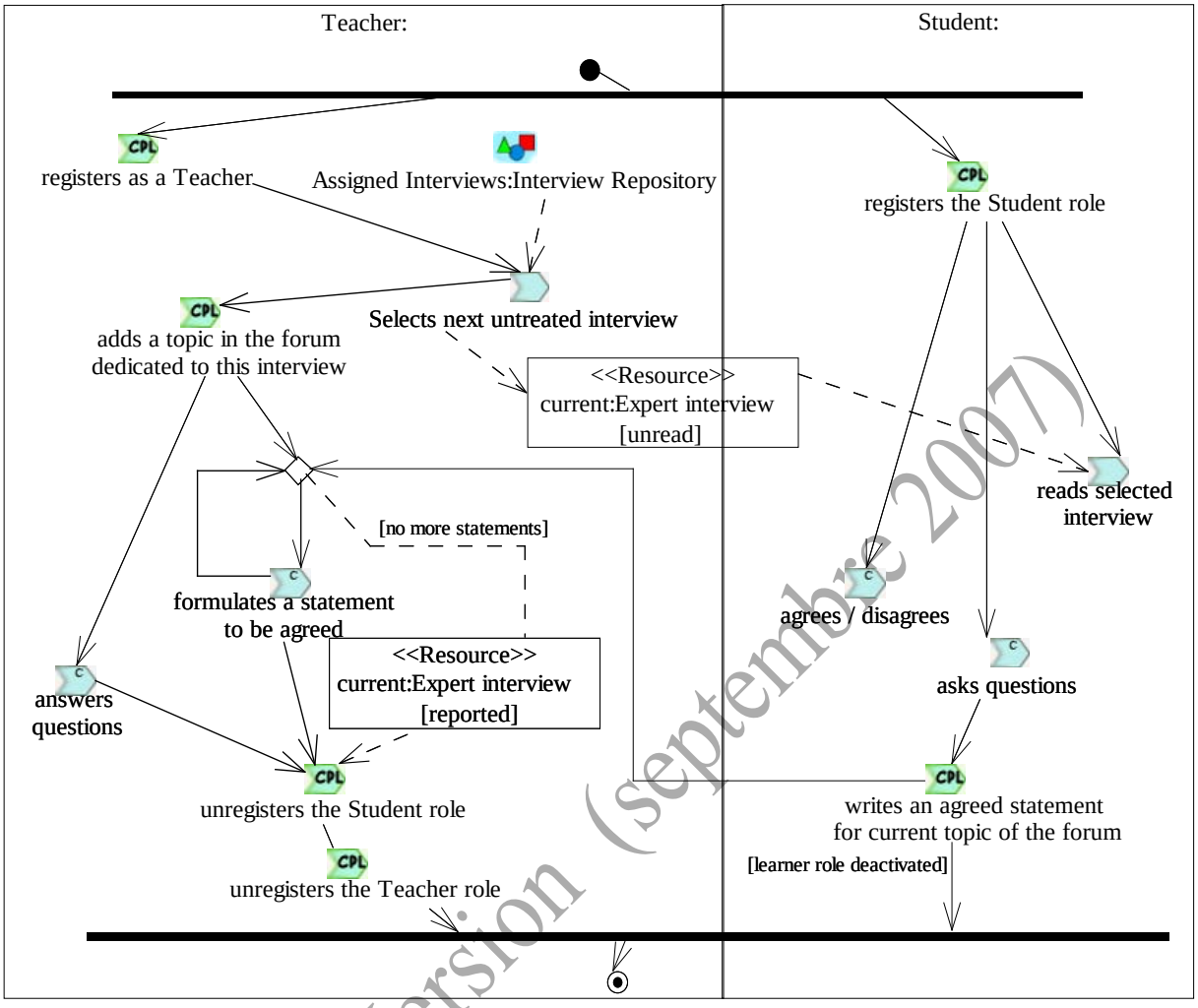


Figure 12: an activity diagram for the reciprocal teaching pattern.

In Figure 13, the <<CPL>> stereotypes denote different functionalities that specialize a forum component: Depending on his role, a concrete actor will register differently; the teacher role has rights to add a topic in the forum while the learner role can write entries for the topic that is currently covered.

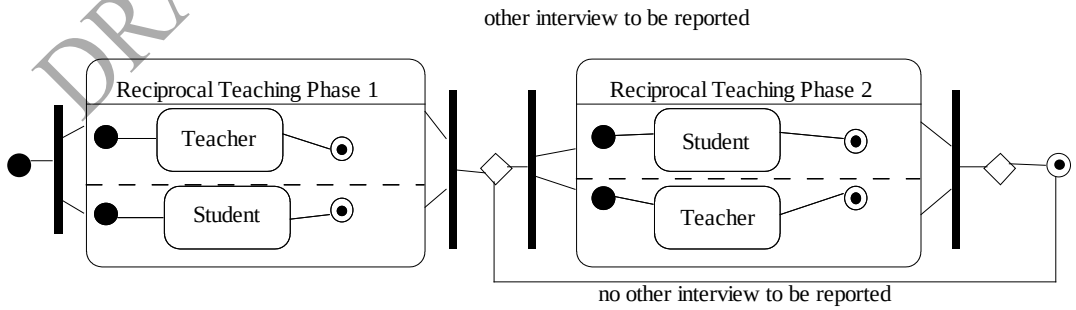


Figure 13: UML state-machine diagram describing both steps of the reciprocal teaching pattern.

Both figures were produced in SMASH PBLs to denote the Reciprocal teaching Pattern (Palincsar and Brown 1986). The term “reciprocal” describes the nature of the interactions each person has in response to the other(s). Teacher and student take turns assuming the role of a dialogue leader (see Figure 13); sequencing of the concrete activities performed by both roles is formalised by the dedicated swimlanes in Figure 12. The ID team chose this pattern

because the designers wanted the students to improve their reading comprehension of the available SMASH testimonies; the designers also wanted them to learn to monitor their own learning and thinking. Thus, in SMASH PBLs, learners' peers are key actors in the reciprocal teaching pattern. These actors successively play the role of the teacher and the role of the student when trying to understand texts or interviews. Figure 12 details how they move from one role to another and what the responsibilities of each role within the collaboration are. For each text (interview), the teacher role has to select one text. The specification states that the teacher role is the one that formulates statements about his reading and understanding but that the student role is the one that can ask questions and which, at the very end of the discussion, will formulate the agreed statements that can be inferred from the reading.

Detailing how such functionalities should be implemented in a specialized forum is outside the scope of CPM. But the layered nature of CPM contributes to the smooth (top-down or bottom-up) transition between the different domains of trust.

Both lessons presented in this section lead us to the following conclusions: even though CPM was specified as a language and not as a design method, experience gained from our case studies enables us to promote a design process in tune with the characteristics of the CPM language. UML is a language; so is CPM. Current object-oriented methods focus on the specification of the static structure of software objects. A noticeable deficiency of these methods is that they do not provide any help on how requirements are refined, how class diagrams can be derived from scenarios, how to specify the active/dynamic parts of a system, or how such a specification may be transformed into an implementation.

During conceptual design, the analysis of the different case studies that we have conducted promotes the idea of bridging the gap between educational needs elicitation (including requirements elicitation, requirements refinement using a combination of use-case diagrams, of activity diagrams, of class diagrams and state-machine diagrams), and the more formal specification of class diagrams which are required to prepare the implementation of a TEL System (Nodenot, Marquesuzaa, Laforcade, & Sallaberry, 2004). The way we used CPM language is as follows. The specification process starts from the definition of use-cases. Each use-case diagram is refined either by other use-case diagrams or by one ore more activity diagrams (representing teaching/learning scenarios). All model elements used in these diagrams are not unrelated parts; they are attributes, messages, etc. which are finally declared in the class diagrams. The behaviour of each class is represented by a set of scenarios (activity diagrams / state machine diagrams) covering the events declared in the specification part of the class.

CONCLUDING REMARKS AND PERSPECTIVES

CPM is a visual, layered, semi-formal, multiple perspectives language dedicated to the description of collaborative learning scenarios with special emphasis on Problem-Based Learning (PBL). By means of the layering mechanism, designers may more easily tackle a complex situation using this graphical and conceptual feature: they start with a coarse-grained description to grasp the global situation and can then decompose each element to get a complete and detailed description (Lesson U2 Observation 1). Next, with the multiple perspectives mechanism, the designers may focus on the sequencing of activities, the behaviour of a particular activity, role responsibilities, *etc*. A complex situation may be described through a set of simple and clear views (lesson U1 Observation 2). Equally, the combination of these two mechanisms promotes collaboration within a team of designers (Lesson U2 Observation 2). Finally, as CPM is dedicated to a specific type of learning

situations, it allows the designer using it to be more likely to be able to describe such situations more quickly than with more general educational languages like IMS-LD. According to model-driven approaches like OMG-MDA (OMG, 2003), these specialized (but limited) languages offer conceptual frameworks for preliminary analysis of learning situations before transforming the resulting models into more operational languages.

Lessons presented in this chapter also reveal some possible ways to improve CPM.

Improvements of CPM

Computer support for design processes

Modelling learning situations is not an easy or usual task for practitioners. Among the several reasons that account for this, we might mention the two most obvious ones. First, practitioners seek to adapt their courses to learners *in situ*, as events occur happen (opportunistic approach) and they tend to prefer to think in terms of content and coarse-grained activities. Second, in educational sciences, models are driven by learning events to detect and to react upon, rather than by a mere sequence of activities which are more typical within the computer world (i.e., workflow sub-domain). So practitioners are not used to getting involved in highly structured course modelling in their everyday routine. Because we are aware of this, we have already proposed guidelines related to CPM through 'best-practices' (Lesson U1 Observation 2) and a design process (lesson U2 Observation 3) in order to help practitioners. (Kent, 2002) already pointed out the problem of 'how to define a model'. He defines it as a main hindrance for the emerging Model Driven Engineering trend. While he generally highlights work about macro-processes ('the order in which models are produced and how they are coordinated'), Kent affirms the need for the MDE community to work on micro-processes, that is to say 'guidelines for producing a particular model'. We consider therefore that we need to improve *CPM micro-processes*. A related perspective must be to provide a computer support for our guidelines. First, such a support will make the application of guidelines easier (and accelerates it). Next, it limits the occurrence of errors caused by the misinterpretation of guidelines. Finally assisting the definition of a model allows designers to learn guidelines in a better way than by only reading the related document.

We have already worked on the computer support for a method dedicated to IMS-LD (Le Pallec, Moura, Marvie, Nebut, & Tarby, 2006). We intend to transpose this previous work to CPM.

Templates

Starting from scratch is another barrier to practitioners when defining models. The *Objecteering* repository provides a way of reusing existing and approved fragments of CPM models (i.e., Figure 12). UML templates address this issue much better. A UML template is a set of parameters to be applied to model elements before use. Such models have the advantage of clearly rendering explicit both the fixed part and the changing part of a model. Equally, defining a template is driven by reusability and modularity which is not the case when defining new model elements duplicated with copy/cut/paste. The application field of a template is consequently broader. However using this mechanism, particularly when defining a template, is not an easy task, especially for a non-UML specialist. Even if *Objecteering* may provide a UML template mechanism, future work will likely involve embedding it into a more user-friendly interface.

Model transformations

The different CPM perspectives are not entirely bound together. The attribute *Testimonies analysis* of *Investigator role 4* (i.e., Figure 10) is not automatically but manually 'deduced' from the link *performs* between *Investigator role 4* and *Testimonies analysis* (i.e., Figure 7). If the link *performs* is removed, the previous attribute will not be automatically removed. Not to impose constraints about the ubiquity of model elements can provide much freedom, and hence flexibility while defining models, especially for practitioners. But in addition to being a source of mistakes, it does not render explicit the repercussion of each action which the designer is performing. To address these two problems, we might consider, for example, developing dynamic transformations between all perspectives so that each action from a perspective should induce logic repercussion on other perspectives. These transformations would be proposed to designers through clickable operations.

Towards other conceptual frameworks

UML and its profile mechanism offers a framework which may prove quickly efficient. First it provides several types of diagrams which enable many aspects to be described. Second, several design processes have emerged from the UML community over the last decade. They describe best-practices related to navigation between previous types of diagrams. Nevertheless, there are some weaknesses. First, defining new modelling concepts with a UML profile requires using (through inheritance) existing UML metaclasses like *Class* or *Actor*. UML profile designers do not necessarily need all inherited attributes or methods. They have to block access to these undesirable properties both in conceptual and graphical ways to respect the semantic of the language they are designing (this can be achieved through OCL constraints or through *J* code (Objecteering, 2006) in case *Objecteering* is used). It is a complex and tedious process if we consider the definition of graphical languages for complex, condensed and non-software engineering metamodels (like IMS-LD). In addition, an efficient profile (that is to say, with conceptual and graphical filtered accesses) generally works only with the UML tool used to define it.

Last, for the time being, it is difficult to provide practitioners with a totally free UML-based model editor given that UML efficient tools are still expensive. Moreover using a UML profile means requiring the use of a whole software engineering oriented environment which may constitute a handicap for practitioners.

It is therefore important to explore alternatives like OMG-MOF (OMG, 2007) or Eclipse/EMF (EMF, 2007) environments. Based on a meta-modelling approach, they present some advantages. For example, defining a language starts with defining a metamodel (abstract syntax) which is not created from existing concepts but from scratch. So there is no need to filter access to model elements because of undesirable inherited features. Another useful functionality of EMF is that creating a metamodel may be done simply by analyzing an XML schema or a DTD. Additionally, there are currently powerful graphical tools like TopCaseD (Farail et al., 2006) and the forthcoming GMF (GMF, 2007) which both allow defining an efficient graphical syntax for a language (concrete syntax). There are of course other facilities which are not as efficient in the UML community, like model transformation engines (GMT for Eclipse (GMT, 2007), YATL for MOF-based models (Patrascoiu, 2004)) and code generation engines like JET (JET, 2007).

But we believe it is still very important to see beyond technology and to maintain a global awareness of how organizational, social and technical issues are impinging on the usability of VIDL.

BIBLIOGRAPHY

Allert, H. (2004). Coherent Social Systems for Learning: an Approach for Contextualized and Community-Centred Metadata. *Journal of Interactive Media in Education, 2*.

Allert, H. (2005). *Modeling Coherent Social Systems for Learning.* Thesis Dissertation, Hannover University (Germany).

Berggren, A., Burgos, D., Fontana, J. M., Hinkelman, D., Hung, V., Hursh, A., et al. (2005). Practical and Pedagogical Issues for Teacher Adoption of IMS Learning Design Standards in Moodle LMS. *Journal of Interactive Media*.

Botturi, L. (2003). *E2ML: Educational Environment Modeling Language.* Ph.D. in Communication Sciences, University of Lugano (Italy).

Botturi, L., Cantoni, L., Lepori, B., & Tardini, S. (2006). *Fast Prototyping as a Communication Catalyst for E-Learning Design*: Making the Transition to E-Learning: Strategies and Issues. Hershey, PA: Idea Group, M. Bullen & D. Janes editors.

Botturi, L., Derntl, M., Boot, E., & Gigl, K. (2006). *A Classification Framework for Educational Modeling Languages in Instructional Design.* Paper presented at the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade (The Netherlands).

Brusilovsky, P. (2004). *KnowledgeTree: a distributed architecture for adaptive e-learning.* Paper presented at the 13th International World Wide Web Conference on Alternate Track Papers, New York (USA).

CopperAuthor. (2005). *CopperAuthor Learning Design editor*, retrieved October 27, 2005 from http://sourceforge.net/projects/copperauthor/

Costagliola, G., De Lucia, A., Orefice, S., & Polese, G. (2002). A classification framework for the design of visual languages. *Journal of Visual Languages and Computing, vol. 13*, pp. 573-600.

Dalziel, J. R. (2006). *Lessons from LAMS for IMS Learning Design.* Paper presented at the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade (The Netherlands).

Derntl, M., & Hummel, K. (2005). *Modeling Context-Aware e-Learning Scenarios.* Paper presented at the 3rd International Conference on Pervasive Computing and Communications Workshops, Kauai Island (Hawai).

Derntl, M., & Motschnig-Pitrik, R. (2007). coUML – A Visual Modeling Language for Cooperative Environments. In L. Botturi & T. Stubbs (Eds.), *Handbook of Visual Languages in Instructional Design; Theories and Practice (in Press)*: Hershey, PA:IDEA Group.

Develay, M. (Ed.). (1993). *De l'apprentissage à l'enseignement*: Collection Pédagogies, ESF Edition.

EMF. (2007). *Eclipse Modeling Framework Project*. Retrieved February 2007, from http://www.eclipse.org/modeling/

Farail, P., Gaufillet, P., Canals, A., Le Camus, C., Sciamma, D., Michel, P., et al. (2006). *The TOPCASED project: a Toolkit in Open source for Critical Aeronautic SystEms Design.* Paper presented at the Eclipse Technology eXchange workshop (eTX) at ECOOP 2006, Nantes (France).

Ferruci, F., Tortora, G., & Vitello, G. (2002). *Exploiting Visual Languages in Software Engineering*: Handbook of Software ENgineering and Knowledge Engineering, Singapore World Scientific Publishing Company.

Fowler, M. (2005). *UML Distilled: Third Edition*: Addison-Wesley.

GMF. (2007). *GMF Project*. Retrieved February 2007, from http://www.eclipse.org/gmf/

GMT. (2007). *GMT Project*. Retrieved February 2007, from http://www.eclipse.org/gmt/

Hernández-Leo, D., Villasclaras-Fernández, E. D., Asensio-Pérez, J. I., Dimitriadis, Y., Jorrín-Abellán, I. M., Ruiz-Requies, I., et al. (2006). COLLAGE: A collaborative Learning Design editor based on patterns. *Educational Technology & Society, 9*(1), pp 58-71.

Hummel, H., Manderveld, J., Tattersall, C., & Koper, R. (2004). Educational modelling language and learning design: new opportunities for instructional reusability and personalised learning. *International Journal of Learning Technology, vol. 1*(1), pp. 111-126.

IMS. (2003a). *IMS Learning Design Best Practice and Implementation Guide*: MS Global Learning Consortium.

IMS. (2003b). *IMS Learning Design Information Model*: IMS Global Learning Consortium.

JET. (2007). *JET Project*. Retrieved February 2007, from http://www.eclipse.org/emft/projects/jet/

Kent, S. (2002). *Model Driven Engineering.* Paper presented at the Third International Conference on Integrated Formal Method, IFM 2002, Turku (Finland).

Koper, R. (2001). *Modeling Units of Study from a Pedagogical Perspective : the pedagogical meta-model behind EML*: Educational Expertise Technology Centre, Open University of the Netherlands.

Koper, R. (2006). Current Research in Learning Design. *Educational Technology & Society, 9(1),* pp 13-22.

Koper, R., Giesbers, K., Van Rosmalen, P., Tattersall, C., Sloep, P. B., Van Bruggen, J., et al. (Eds.). (2003). *A Design Model for Lifelong Learning Networks*.

Koper, R., & Olivier, B. (2004). Representing the Learning Design of Units of Learning. *Educational Technology and Society, 7*(3), 97-111.

Laforcade, P. (2004). *Méta-modélisation UML pour la mise en oeuvre de situations problèmes coopératives.* Doctorat en informatique de l'Université de Pau et des Pays de l'Adour (France).

Le Pallec, X., Moura, O., Marvie, R., Nebut, M., & Tarby, J.-C. (2006). *Supporting generic methodologies to assist IMS-LD modelling.* Paper presented at the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade (Netherlands).

Meirieu, P. (Ed.). (1994). *Apprendre... Oui mais comment?* : ESF Edition.

Miao, Y. (2000). *Design and implementation of a Collaborative Virtual Problem-Based Learning Environment.* Thesis of the University of Darmstadt (Germany).

Nodenot, T. (2005). *Contribution à l'Ingénierie dirigée par les modèles en EIAH : le cas des situations-problèmes coopératives.* Habilitation à diriger les recherches en Informatique de l'Université de Pau et des Pays de l'Adour (France).

Nodenot, T., & Laforcade, P. (2006). A game about planets: Elements for a Didactical transposition described with the CPM language, *Paper presented during the Workshop entitled "Comparing Educational Modeling Languages on a case study", 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*. Kerkrade (The Netherlands).

Nodenot, T., Marquesuzaà, C., Laforcade, P., & Sallaberry, C. (2004, 17-22 may 2004). *Model based Engineering of Learning Situations for Adaptive Web Based Educational Systems.* Paper presented at the ACM Thirteenth International World Wide Web Conference (IW3C2 Conference), ISBN : ACM 1-58113-912-8/04/0005, New-York (USA).

Norman, D. A., & Spohrer, H. G. (1996). Learner-centered Education. *Communication of the ACM, volume 39, n°4.*

Objecteering. (2006). *Objecteering J Language User Guide*. Retrieved February 2007, from http://www.objecteering.com/doc/j_language/toc.htm

OMG. (1999). *White Paper on the Profile Mechanism*: Object Management Group Analysis and Design Task Force.

OMG. (2002a). *The Software Process Engineering Management Metamodel (SPEM)*: Technical Report formal/2002-11-14.

OMG. (2002b, Août 2002). *UML 2.0 Superstructure Specification*, from http://www.omg.org/docs/ptc/03-08-02.pdf

OMG. (2003). *MDA Guide Version 1.0.1*, from http://www.omg.org/docs/omg/03-06-01.pdf

OMG. (2007). *Meta-Object Facility (MOF) Specification*. Retrieved February 2007, from http://www.omg.org/mof/

Paramythis, A., & Loidl-Reisinger, S. (2004). Adaptive Learning Environments and e-Learning Standards. *Electronic Journal of e-Learning, vol. 2*(2).

Patrascoiu, O. (2004). *YATL:Yet Another Transformation Language*. Paper presented at the 1st European MDA Workshop, University of Twente (The Netherlands).

Pawlowski, J. (2002). *Reusable Models of Pedagogical concepts: a Framework for Pedagogical and Content Design.* Paper presented at the International Conference ED-MEDIA 20002, Denver (USA).

Pawlowski, J., & Bick, M. (2006). Managing and re-using didactical expertise: The Didactical Object Model. *Educational Technology and Society, 9*(1), 84-96.

Reload. (2005). *Reusable eLearning Object Authoring & Delivery Project*, October 27, 2005 from http://www.reload.ac.uk/.

Renaux, E., Caron, P.-A., & Le Pallec, X. (2005). *Learning Management System component-based design: a model driven approach.* Paper presented at the Montreal Conference on e-Technologies (Mcetech), Montréal (Canada).

Sallaberry, C., Nodenot, T., Marquesuzaà, C., Bessagnet, M.-N., & Laforcade, P. (2002). *Information modelling within a Net-Learning Environment.* Paper presented at the 12th Conference on Information Modelling and Knowledge Bases, Krippen, Swiss Saxony (Deutschland).

Sampson, D., Karampiperis, P., & Zervas, P. (2005). ASK-LDT: A Web-Based Learning Scenarios Authoring Environment based on IMS Learning Design. *International Journal on Advanced Technology for Learning (ATL), 2*(4), pp. 207-215.

Santos, O. C., Boticario, J. G., & Barrera, C. (2005). *aLFanet: An adaptive and standard-based learning environment built upon dotLRN and other open source developments.* Paper presented at the 2005 dotLRN conference, Madrid (Spain).

Schneemayer, G. (2002). *Contextual Web Services for Teaching*. Ludwig Maximilians Universität, München (Allemagne).

Stahl, G. (2006). *Group Cognition: Computer Support for Building Collaborative Knowledge*: Cambridge, MA: MIT Press.

Tattersall, C. (2007). Using the IMS Learning Design notation for the modelling and delivery of education. In L. Botturi & T. Stubbs (Eds.), *Handbook of Visual Languages for Instructional Design: Theories and Practices*: IDEA Group Inc.

Vignollet, L., David, J.-P., Ferraris, C., Martel, C., & Lejeune, A. (2006). Comparing Educational Modeling Languages on a Case Study, *Workshop in conjunction with the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*. Kerkrade (The Netherlands).

Vogten, H., & Martens, H. (2003). *CopperCore – The IMS Learning Design Engine*, retrieved October 27, 2005 from http://www.coppercore.org.